

LA-UR-18-23001

Approved for public release; distribution is unlimited.

Title: Charliecloud: Unprivileged Containers for HPC

Author(s): Jennings, Michael E.
Priedhorsky, Reid
Randles, Timothy C.

Intended for: HPCXXL 2018 Winter Meeting / HPC Advisory Council 2018 Swiss
Conference, 2018-04-09 (Lugano, Switzerland)

Issued: 2019-04-17 (rev.1)

Disclaimer:

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy under contract 89233218CNA000001. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.



Unprivileged Containers for HPC

Michael Jennings ([@mej0](#)) mej@lanl.gov

Reid Priedhorsky reidpr@lanl.gov

Tim Randles trandles@lanl.gov

2018 HPCXXL Winter Meeting
2018 HPC Advisory Council Swiss Conference
Lugano, Switzerland

UNCLASSIFIED



Los Alamos National Laboratory



- Established in 1943 as “Site Y” of the Manhattan Project to create atomic bomb
 - Mission: To solve National Security challenges through Scientific Excellence
 - Part of the NNSA “Tri-Lab” partnership with Lawrence Livermore and Sandia Labs
 - We perform a wide variety of classified and open scientific research and development.
- Funded primarily by the Department of Energy, we also do extensive work for/with the Departments of Defense and Homeland Security, the Intelligence Community, et al.
 - Our strategy reflects US government priorities including nuclear security, intelligence, defense, emergency response, nonproliferation, counterterrorism, and more.
 - We help to ensure the safety, security, and effectiveness of the US nuclear stockpile.
 - Since 1992, the United States no longer performs full-scale testing of nuclear weapons. This has necessitated continuous, ongoing leadership in large-scale simulation capabilities realized through investment in high-performance computing.

UNCLASSIFIED



Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA



UNCLASSIFIED | 2

LANL HPC Division

- LANL's history in high-performance computing is long and storied, dating back to the early '50s.
- Accomplishments include:
 - Helped IBM develop the 1st transistor-based supercomputer, Stretch
 - Our CM-5 was #1 on the inaugural Top500 List
 - The 1st vector computer, Cray-1, deployed here
 - 1st hybrid supercomputer (using IBM POWER and PlayStation Cell processors), Roadrunner, was also 1st to break the PetaFLOP/s barrier
- Led by Gary Grider, creator of Burst Buffer technology
- We support over 2000 unique users across more than 100 different classified/open science projects on 20+ clusters



UNCLASSIFIED

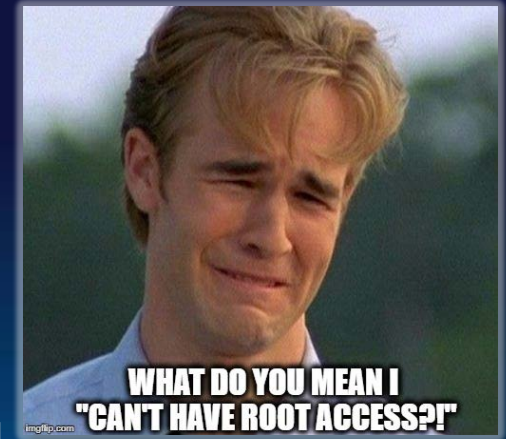
First World Problems in HPC

Problem #1: HPC clusters have narrowly focused software stacks.

- They do serial and parallel-MPI tasks well...but that's it.
- Compute node images are often in RAM/NFS & kept small.
- Managing multiple OSs, and required expertise, is rare & labor intensive.

Problem #2: We're old. And not getting any younger.

- Schools are teaching clusters, "parallel"/scalable/distributed software development...but usually "embarrassingly parallel" (e.g., map/reduce)
- All the "cool kids" are using Ubuntu (or Arch, or Alpine...), not RHEL
- Modern Machine Learning and Data Analytics toolsuites are non-trivial



Problem #3: We are finite, as is our time.

- We generally won't install extra software with low user demand.
- Unique or unusual use cases tend to be lower priority.
- The line between "innovator" and "crackpot" is often rather blurry....

UNCLASSIFIED

The Solution: UDSS/UDI/BYOE

User-Defined Software Stacks (UDSS) allow users to supply not only their own applications/source to run on HPC systems but also the environment -- up to and including entire OS images -- in which they should run!

Advantages include portability, usability, consistency, time savings...

Potential disadvantages include missing functionality (HSN, accelerators, filesystems), performance degradation; thus, addressing these should be part of the design of any HPC-focused solution!

In rare, specific cases, certain packages may address this independently by building static binaries or otherwise coupling dependencies with executables

- Works everywhere
- No privileges required
- Requires build-time support
- May or may not be feasible
- For everyone else, our options are...



UNCLASSIFIED

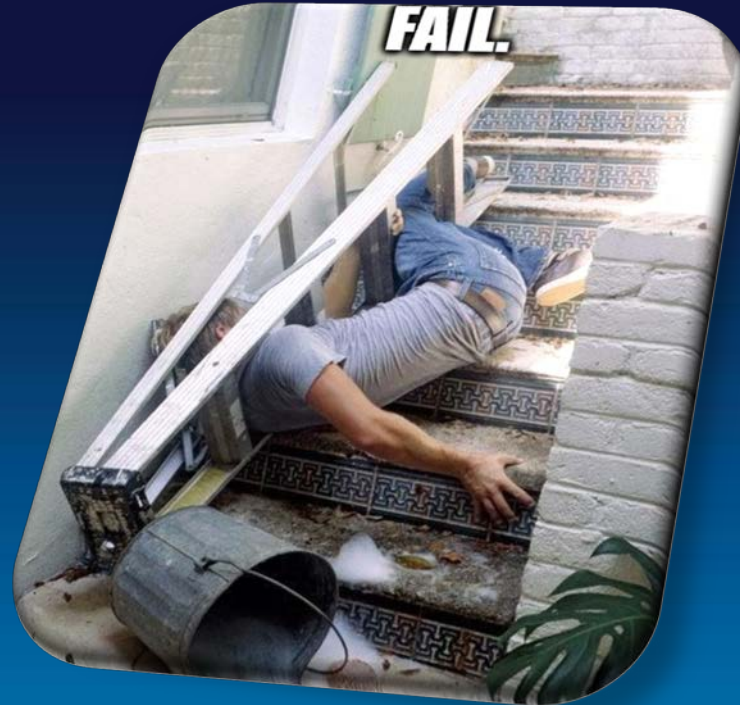
Option #1: Compile It Yourself

Advantages

- Available everywhere immediately
- Requires no privileges
- No additional privacy/security risk
- Direct access to all hardware
- No performance penalty
- Theoretically applies to any open source software library/app/stack

Disadvantages

- It's the 21st Century; we no longer *want* to party like it's 1999!
- Tedious and time-consuming
- Error-prone with exponential propagation of errors/time-to-fix
- Hard to update; start over each time
- No standard workflow/reproducibility
- Provides neither portability nor consistency



Nope.

UNCLASSIFIED

Option #2: Environment Manager

Advantages

- Available for most IA32/x64 systems
- Requires no privileges
- No additional privacy/security risk
- Direct access to all hardware
- No performance penalty

Disadvantages

- Frequently still requires building from source (time/space constraints)
- Varying degrees of HPC support
- Varying degrees of reproducibility, portability, consistency, workflow
- Users/consultants bear entire burden

If You Must...

- Good options include EasyBuild, Lmod, Spack
- Also Anaconda, nixOS



*“There’s GOT to be a
BETTER WAY!!”*

UNCLASSIFIED

Option #3: Virtualization (VMs)

Advantages

- Ultra-flexible (any kernel/OS/arch)
- Strong-to-complete isolation
- Common use cases perform well

Disadvantages

- Performance suffers for most HPC use cases, often significantly
- Performance/isolation tradeoffs
- Infrastructure can be complex
- Direct/performant access to hardware may require privileges
- Not all “exotic” HPC hardware supported
- Entire OS must be provisioned and booted, separate hostname/IP, etc.



Should HPC become the Cloud?

UNCLASSIFIED

Option #4: Containers

Advantages

- Enough flexibility (only share kernel)
- Enough isolation (namespaces, etc.)
- Standard, reproducible workflow
- Bare-metal performance (or close)
- Minimal or no user/consultant burden (presumes correct solution choice)
- Extremely simple and easy-to-use (presumes correct solution choice)

Disadvantages

- Require recent Linux kernel/distro (SLES 12SP2, RHEL 7.4, Ubuntu 16.04, Linux LTS 4.4/4.9) **or** privilege
- Occasional growing pains due to newness (feature-complete in 2013)
- Container expertise in HPC still rare; thus, bad/misleading info & myths are ubiquitous (even in publications)!



Should HPC use containers?

UNCLASSIFIED

So...Umm...These Container Things...Like, What Are They?

Good question! Not everyone agrees. Here's our take.

Linux Containers:

- Use one or more kernel **namespaces** to provide isolation for (i.e., “contain”) a process (along with its child processes, if any);
- Envelope/restrict the process(es) such that escape/escalation is “impossible;” and
- Facilitate application security by providing capability constraints, integrity assurance, and content validation as required via industry-standard formats and workflows.



UNCLASSIFIED

Namespaces, You Say?

The Linux Kernel supports 6 namespaces as of version 3.8, 7 as of 4.6.

- 6 Privileged Namespaces (require `CAP_SYS_ADMIN` to create)
 - `mount` – Private filesystem mount points, recursion/propagation controls
 - `pid` – Private view of process IDs and processes, `init` semantics
 - `uts` – Private hostname and domainname values
 - `net` – Private network resources (devices, IPs, routes, ports, etc.)
 - `ipc` – Private IPC resources (SysV IPC objects, POSIX msg queues)
 - `cgroup` – Private control group hierarchy (Linux 4.6+ only)
- 1 Unprivileged Namespace (requires no special capabilities to create)
 - `user` – Private UID and GID mappings
 - Can be combined with other namespaces, even if unprivileged
- System Call API: `unshare(2)`, `clone(2)`, `setns(2)`

Further reading: “Namespaces in Operation” (<https://lwn.net/Articles/531114/>)

UNCLASSIFIED

Additional Container Elements

The Linux kernel has several additional subsystems that containers use:

- **cgroups** – Control hierarchical resource management and constraints
 - Latest kernels (4.6+) even have namespaces for this!
 - This is how modern schedulers/RMs track and control job resource utilization
- **seccomp- bpf** – Berkeley Packet Filter-based system call filtering
 - Frequently used to prevent containers from exceeding their scope
- **prctl (PR_SET_NO_NEW_PRIVS)** – Prevent privilege escalation
 - Permanent, kernel-level flag that prevents `execve()` ever granting privileges
 - Preserved across all calls to `fork()`, `clone()`, and `execve()`
- **SELinux** – MLS/MAC Labeling system for filesystems and applications
 - Allows admins precise control over actions, roles of applications
- **AppArmor** – Profile-based MAC system for limiting applications' abilities
 - Similar to SELinux but without filesystem labeling features

UNCLASSIFIED

The Container Landscape

Full-featured Container Systems

- Support building, distribution, validation, and execution
- Provide for complete handling of containers throughout lifecycle
- Examples: Docker, CRI-O, LXC/LXD
- Most modern container engines now implement Open Container Initiative (OCI) Image and/or Runtime Spec(s)
- Building containers is still a per-system function. OCI does NOT “do” building!
 - Dockerfiles are the de facto standard; robust, capable DSL
 - CoreOS Rocket supplied a build based around traditional shell-fu
 - ProjectAtomic buildah does both!

Lightweight Container Systems

- Generally only provide runtime; most leverage Docker/OCI/K8s ecosystem
- Tend to require existing directory tree (i.e., flattened image) to run in
- Examples: RunC, CCon, NsJail, unshare(1), systemd-nspawn(1)
- ...and of course, Charliecloud!



UNCLASSIFIED

Let's Talk Security

Modern Cybersecurity requires many layers; true security isn't merely the absence of access but rather the presence of protection!

- Prior to invocation, container security relies on a wide variety of content assurance methodologies (e.g., non-repudiation, CAS) to insure end-to-end transport safety.
- At runtime, container security relies entirely on the separation of authorities/roles within the Linux kernel (e.g., seccomp, MLS). **If the kernel fails, the container fails.**
- Lightweight container solutions have the advantage of leaving much of that to others.
- Most mindshare is in Linux kernel hardening and native unprivileged containers.



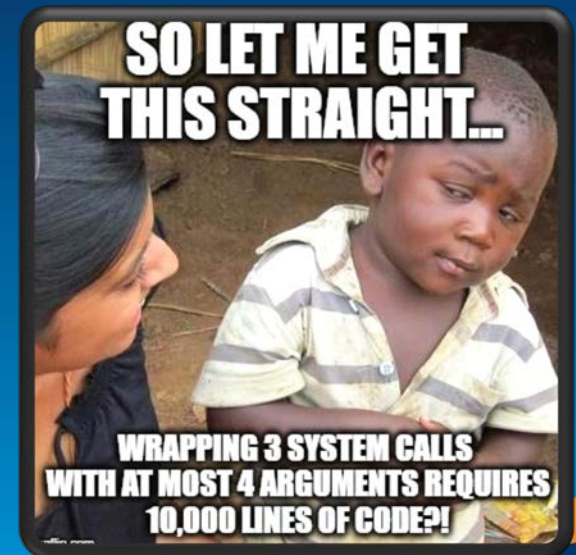
All container solutions expose (formerly) privileged operations to unprivileged users, requiring us to accept a new security boundary! So, do you want:

- *one that's new and known to be flawed?*
- *a well-understood one that still relies on setuid 0?*
- *the one that provides all relevant information to the kernel regarding separation of privilege?*

UNCLASSIFIED

Why Charliecloud?

- Docker/Moby aren't (yet?) a good fit for HPC:
 - Performance - OverlayFS is sloooooooooow due to layering
 - Integrity - File removal is done by "whiting out" which causes certain subtle issues
 - Associativity - Docker containers are children of daemon, not CLI
- NERSC's Shifter offers appealing user experience but has overhead
 - User downloads container(s) from Docker Hub, submits jobs to run in them
 - Admin must set up & maintain Image Gateway
 - Relies primarily on `chroot()` ; requires `setuid-root`
- Small/simple APIs should yield small/simple code!
 - Charliecloud weighs in at ***fewer than 1000*** LoC
 - Compare to NSJail (4,000), Shifter (19,000), Singularity (15,000), and Docker (160,000)!
- Charliecloud's security boundary **is** the kernel
 - Also worth watching: Moby, Rootless RunC, CRI-O



UNCLASSIFIED

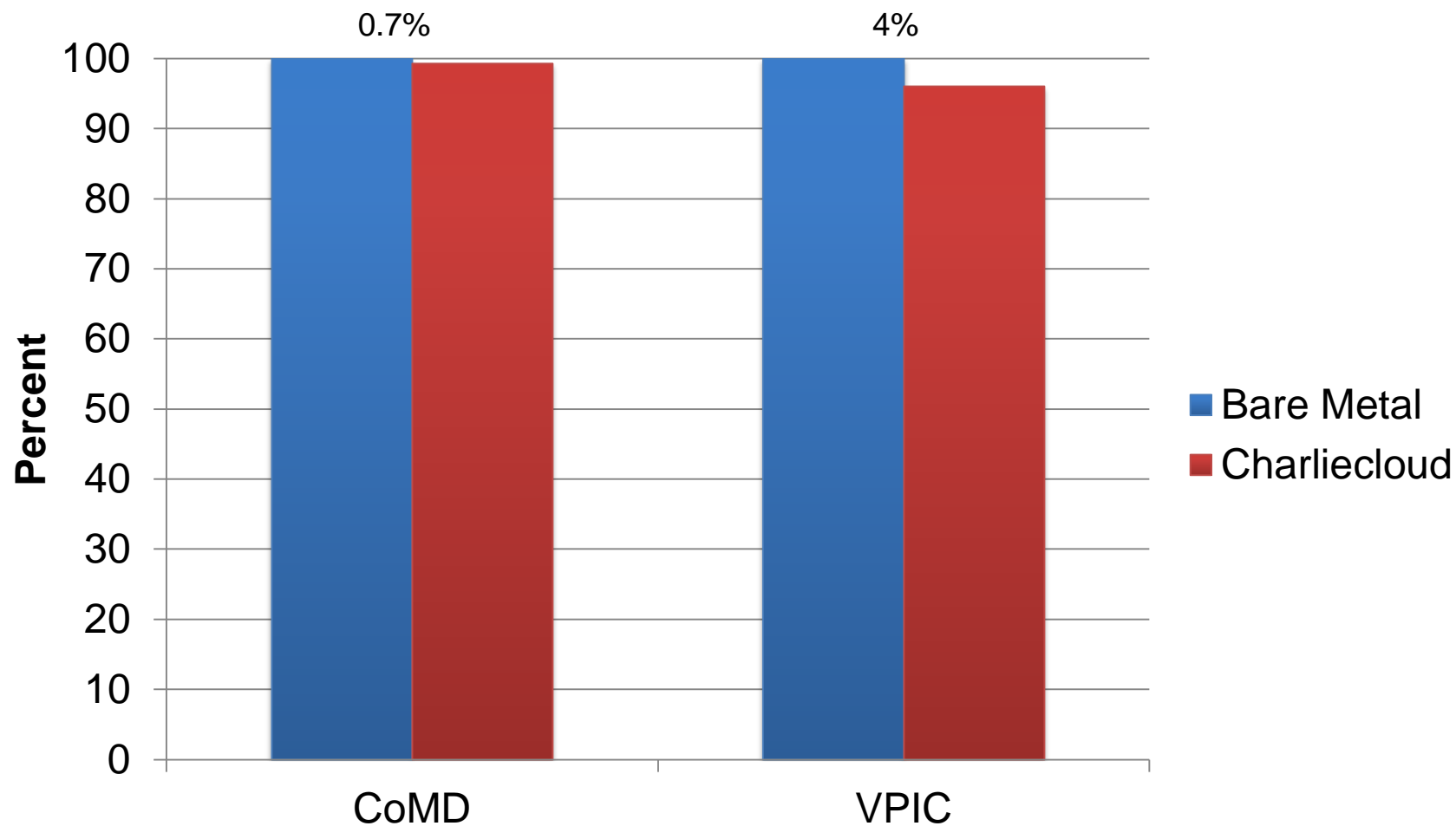
Charliecloud Demo/Walkthrough

```
$ cd ~/charliecloud/examples/serial/hello
$ ls
Dockerfile  hello.sh  README  test.bats
$ ch-build -t hello ~/charliecloud
Sending build context to Docker daemon 15.19 MB
[...]
Successfully built 30662b3f94f3
$ ch-docker2tar hello /var/tmp
57M /var/tmp/hello.tar.gz
```

```
$ ch-tar2dir /var/tmp/hello.tar.gz /var/tmp/hello
creating new image /var/tmp/hello
/var/tmp/hello unpacked ok
$ ch-run /var/tmp/hello -- cat /etc/debian_version
8.9
```

UNCLASSIFIED

Charliecloud MPI Performance



UNCLASSIFIED

Charliecloud Resources



- Supercomputing 2017 Paper
by Reid Priedhorsky and Tim Randles
 - “Charliecloud: Unprivileged Containers for UDSS in HPC”
 - Los Alamos Tech Report LA-UR-17-30438
 - <http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-17-30438>
- ;login: Article “Linux Containers for Fun & Profit in HPC”
by Reid Priedhorsky
 - <https://www.usenix.org/publications/login/fall2017/priedhorsky>
- Documentation: <https://hpc.github.io/charliecloud> (includes tutorials!)
- Source Code: <https://github.com/hpc/charliecloud>
- Mailing List: charliecloud@groups.io || <https://groups.io/charliecloud>
- Contact Reid (reidpr@lanl.gov), Tim (trandles@lanl.gov),
or Michael (mej@lanl.gov, [@mej0](https://twitter.com/mej0) on Twitter)

UNCLASSIFIED



Questions/Comments?

UNCLASSIFIED

THANK YOU!

PS: These slides, and all unclassified LANL publications, are available via LA-UR number at:
<http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-18-23001>

UNCLASSIFIED